In Understanding Language Understanding: Computational Models of Reading, edited by Ashwin Ram and Kenneth Moorman, The MIT Press.

Information Extraction as a Stepping Stone toward Story Understanding

Ellen Riloff
Department of Computer Science
University of Utah
Salt Lake City, UT 84112
riloff@cs.utah.edu

Historically, story understanding systems have depended on a great deal of hand-crafted knowledge. Natural language understanding systems that use conceptual knowledge structures [SA77, Cul78, Wil78, Car79, Leh81, Kol83] typically rely on enormous amounts of manual knowledge engineering. While much of the work on conceptual knowledge structures has been hailed as pioneering research in cognitive modeling and narrative understanding, from a practical perspective it has also been viewed with skepticism because of the underlying knowledge engineering bottleneck. The thought of building a large-scale conceptual natural language processing (NLP) system that can understand open-ended text is daunting even to the most ardent enthusiasts. So must we grit our collective teeth and assume that story understanding will be limited to prototype systems in the foreseeable future? Or will conceptual natural language processing ultimately depend on a massive, broad-scale manual knowledge engineering effort, such as CYC [LPS86]?

Perhaps the answer lies in the current trends of *information extraction* research. Information extraction (IE) is a form of natural language processing in which certain types of information must be recognized and extracted from text. Although IE systems are typically designed for a single domain, there is a great deal of interest in building systems that are easily portable to new domains. Many researchers are developing methods to acquire the necessary domain-specific knowledge automatically. The goal is not necessarily to produce a general-purpose information extraction system, but to create tools that would allow users to build customized information extraction systems quickly.

If the IE researchers are ultimately successful, we could imagine being able to create, quite literally, an IE system "du jour" that is tailored to our interests on any given day. Another possible future scenario is to assemble a large suite of IE systems, each being a specialist in extracting information pertaining to its own area of expertise. Collectively, the suite of systems could support conceptual natural language understanding capabilities, if not for every subject then at least for a wide variety of subjects.

On the surface, information extraction might appear to be fundamentally different from story understanding. And there are some important differences between these tasks. But we will argue that the challenges and difficulties underlying them are largely the same. A corollary of this view is that automated knowledge acquisition techniques developed for information extraction are likely to be applicable to story understanding as well. In this chapter, we will explain how research in information extraction may be a significant stepping stone toward progress in story understanding. We will discuss emerging trends in the information extraction community, draw parallels between the two types of natural language understanding, and describe recent research in automated case frame generation for information extraction.

1 Information Extraction

1.1 What is information extraction?

Information extraction is a subfield of natural language processing that is concerned with identifying predefined types of information from text. For example, an information extraction system designed for a terrorism domain might extract the names of perpetrators, victims, physical targets, weapons, dates, and locations of terrorist events. Or an information extraction system designed for a business domain might extract the names of companies, products, facilities, and financial figures associated with business activities.

Natural language understanding is crucial for most information extraction tasks because the desired information can only be identified by recognizing conceptual roles. We use the term "conceptual role" to refer to semantic relationships that are <u>defined</u> by the role that an item plays in context. For example, extracting noun phrases that refer to people can be done without regard to context by searching for person names, titles, and personal pronouns, such as "Mary," "John," "Smith," "Mr.," "she," and "him." Contextual information may be necessary for word sense disambiguation (for example, distinguishing the person "John Hancock" from the company "John Hancock"), but that is a separate issue.

In contrast, some semantic relationships are defined by the role that an item plays in a larger concept. For example, "perpetrator" and "victim" are conceptual roles related to the concept of crime. One cannot identify perpetrators and victims simply by looking at names. "John Doe" could be a perpetrator in one context but a victim in another. In both cases "John Doe" is a person, but his role as perpetrator or victim depends entirely on the surrounding context.

Similarly, one can identify many companies by searching for known company names such as "IBM," and capitalized noun phrases ending with "Inc.," "Co.," or "Corp." But it is impossible to identify companies involved in a merger simply by searching for company names. The surrounding context determines the conceptual role of the company. For example, "XYZ Corp." might be involved in a merger, a joint venture, an acquisition, or a charity event depending on its role in the larger context. Understanding conceptual roles is essential for these types of problems.[1]

Most information extraction systems use some form of extraction pattern to identify potentially relevant information. For example, the pattern <**subject**> **was bombed** might be used to identify bombing targets. Whenever a passive form of the verb “bombed” is encountered, this pattern extracts the subject of the verb as a bombing target. Some extraction patterns are more complicated than others, but most extraction patterns can be viewed as simple case frames. Each case frame is activated by specific linguistic expressions and then extracts surrounding phrases as slot fillers.

To illustrate, Figure 1 shows a murder case frame with multiple slots. This case frame is activated by the word “murdered” whenever it appears as a passive verb. The case frame contains three slots to extract a victim, perpetrator, and weapon (instrument). Each item is extracted from a different syntactic constituent in the clause. For example, the subject of the verb is extracted as the murder victim and the object of the preposition “by” is extracted as the perpetrator. Selectional restrictions are often used to check that an extracted item satisfies certain semantic constraints. For example, the %MURDERED% case frame requires victims and perpetrators to be human. Throughout the rest of this article, we will refer to extraction patterns and case frames interchangeably, with the understanding that case frames for other tasks may be significantly more complex.

Name:	%MURDERED%
Event Type:	MURDER
Trigger Word:	murdered
Activating Conditions:	passive-verb
Slots:	VICTIM <subject> (*human*)
	PERPETRATOR <prep-phrase, by> (*human*)
	INSTRUMENT <prep-phrase, with> (*weapon*)

Figure 1: A case frame for information extraction

The components of an information extraction system vary depending on the approach that is used, but most IE systems perform part-of-speech tagging, partial parsing, semantic interpretation, case frame instantiation, and some sort of discourse analysis. Much of the work on information extraction has been fostered through a series of message understanding conferences (MUCs) sponsored by the U.S. government (e.g., see [MUC93, MUC92, MUC91]).

1.2 The message understanding conferences

Since the late 1980’s, the U.S. government has been sponsoring Message Understanding Conferences (MUCs) to evaluate and advance the state-of-the-art in information extraction. In the last few years, these message understanding conferences have grown in participation, scope, and visibility. The MUCs are competitive performance evaluations involving a set of participants from different sites, usually a mix of academic and industrial research labs. Each participating site builds an IE system for a predetermined domain. The IE systems are all evaluated on the same domain and text collection, and the results are scored using an official scoring program developed by the MUC organizers.

The message understanding conferences are especially noteworthy because they represent the first large-scale effort to evaluate natural language processing systems. The question of *how* to evaluate an NLP system is a non-trivial issue, so the development of standard scoring criteria was a worthwhile contribution in its own right. Evaluation is a complicated and sometimes controversial issue, but the MUC scoring program and criteria represents an important first step in confronting this problem.

The MUC meetings are also important as a forum for comparing the performance of different NLP techniques on a uniform task and text collection. Until these conferences, most NLP systems were developed in isolation and designed for their developer's favorite domain. The message understanding conferences have served as a platform for comparing and contrasting different NLP approaches on equal footing. Furthermore, the MUC tasks involve real, unconstrained text — primarily news wire articles. The message understanding conferences have played an important role in pushing the NLP community toward realistic text applications.

One of the most interesting aspects of these conferences has been the evolution of IE systems over the years. Initially, there was a great deal of variation across systems, representing a broad range of natural language processing techniques. But over time, the IE systems converged so that most IE systems today share relatively similar architectures and approaches. Many lessons have been learned about the strengths and weaknesses of various techniques. In the next section, we survey some of the most notable trends in the information extraction community and explain their relationship to issues in story understanding.

2 Trends in Information Extraction

As historians well know, watching the evolution of a field often provides valuable insights into the nature of its problems and possible solutions. So it is useful to take a step back every once in a while and reflect on general directions that have emerged over time. We will discuss three such trends in information extraction: a convergence toward partial parsing, a healthy respect for discourse analysis, and an emphasis on automated knowledge acquisition. It would be presumptuous to claim that these are the only trends that have taken place in the field, but these three are particularly interesting from the perspective of conceptual natural language processing and story understanding.

2.1 Partial parsing

One of the most obvious trends in the IE community has been a convergence toward partial parsing techniques. In 1991, the systems presented at the Third Message Understanding Conference (MUC-3) reflected a wide variety of parsing techniques. Some systems were grounded in linguistic theory and attempted to generate a complete parse tree for each sentence in a text (e.g., [DLW$^+$91, GSM91, MSBS91]). At the other end of the spectrum, the TTS system [DGCN91] did virtually no syntactic analysis at all. The diversity of syntactic approaches was quite remarkable, running the gamut from full syntactic parsing to practically no syntactic parsing and everything in between.

As the field matured, the syntactic components began to look increasingly alike. Today, almost all IE systems use partial parsing techniques. Even the most linguistically-oriented IE researchers have reluctantly accepted that, for the information extraction task at least, the potential benefits of full parsing are usually overwhelmed by the extra overhead and lack of robustness. Real news articles pose major challenges for syntactic parsing because of ungrammatical text, complex and lengthy grammatical constructs, and massive ambiguity.

For information extraction, the information being sought can often be identified by searching a single clause or phrase. The remaining clauses and phrases may be ignored because they do not contain relevant information. For example, consider the sentence below, which is the first sentence of a MUC-3 text.

> IN AN ACTION THAT IS UNPRECEDENTED IN COLOMBIA'S HISTORY OF VIOLENCE, UNIDENTIFIED PERSONS KIDNAPPED 31 PEOPLE IN THE STRIFE-TORN BANANA-GROWING REGION OF URABA, THE ANTIOQUIA GOVERNOR'S OFFICE REPORTED TODAY.

The relevant information in this sentence is that unidentified persons kidnapped 31 people in the region of Uraba. The rest of the sentence can be effectively ignored. Simply looking for the pattern <X> **kidnapped** <Y> **in** <Z> will identify the perpetrators, victims, and location.

For information extraction, generating complete parse trees has had minimal reward thus far. Researchers committed to full syntactic parsing might offer the explanation that information extraction does not require full natural language processing capabilities because only specific types of information need to be recognized. There is some validity to this argument since, by definition, information that is not relevant to the domain can be safely ignored. However, determining which information is relevant is not always easy and simple pattern recognition is not enough. For example, consider the following sentences:

1. The mayor was killed by FMLN guerrillas.
2. The mayor was killed by a burglar.
3. The mayor was killed by armed men.
4. The mayor was killed by armed men belonging to the FMLN.
5. The mayor was killed by armed men during a hold-up in a convenience store.
6. The mayor was killed by armed men in retaliation for the arrest of a prominent FMLN leader.
7. The mayor was killed and the FMLN claimed responsibility for the murder.

In the first 6 sentences, the perpetrator can be extracted using the simple pattern **killed by <X>**. In sentence #1, the word "FMLN" identifies the perpetrators as terrorists so this murder was clearly terrorist in nature. In sentence #2, the word "burglar" suggests a criminal robbery that was probably not terrorist in nature. But the perpetrators in sentences #3,4,5,and 6 are described only as "armed men." Determining whether they are terrorists or not requires information from elsewhere in the sentence. And in sentence #7, there is no specific reference to a perpetrator at all, but the second clause states that a terrorist organization has claimed responsibility for the murder so we can infer that the FMLN is the perpetrator. These sentences illustrate that identifying relevant information can be quite complicated. In many cases, local patterns are not enough and the entire sentence must be understood. And sometimes inferences must be generated across sentences or even paragraphs. Inference can be essential in determining the relevance of an event or a piece of information.

Given these difficulties, one might think that more syntactic analysis is needed. But syntactic analysis has been relegated to a relatively minor role in most information extraction systems. Concept activation and instantiation are the central activities. Syntax is used to map syntactic roles to conceptual roles in case frames, but this is usually accomplished with a fairly shallow syntactic analysis. The domain-specific expressions that activate a case frame and the conceptual roles defined by a case frame are the key elements that identify potentially relevant portions of text. The case frames are arguably the most crucial component of an IE system; without the case frames, no information would be extracted at all.

For most IE tasks, discourse processing is also crucially important to the success of the system. As we will describe in the next section, discourse processing can be especially challenging when the discourse analyzer has only partial information to work with. Full parse trees and deeper syntactic analysis would surely provide more clues to aid discourse analysis. But conceptual analysis, domain knowledge, and memory organization are still at the heart of the discourse problem.

2.2 Discourse analysis

Perhaps one of the most sobering results of the message understanding conferences has been the realization that discourse analysis of written text is a wide open problem. While there has been substantial work on analyzing spoken discourse, there has been much less work on written discourse. Furthermore, most discourse theories depend on large amounts of world knowledge, and have not been tested empirically on large text collections. Real texts are typically more irregular and arbitrary than most discourse theories expect.

In general, discourse processing of written text involves tracking events and objects and understanding the relationships between them. Anaphora resolution is one such problem, which includes resolving pronouns, proper nouns, and definite noun phrases. Topic recognition and segmentation is another key problem. For example, consider the MUC-4 text shown in Figure 2.

This text describes a series of terrorist attacks in protest of the murder of the group's leader, Bernardo Jaramillo Ossa. The first paragraph mentions both the terrorist attacks and Jaramillo's

(1) MEMBERS OF THE 8TH FRONT OF THE SELF-STYLED REVOLUTIONARY ARMED FORCES OF COLOMBIA [FARC] HAVE CARRIED OUT TERRORIST ATTACKS IN SOUTHERN CAUCA DEPARTMENT TO PROTEST PATRIOTIC UNION [UP] PRESIDENTIAL CANDIDATE BERNARDO JARAMILLO OSSA'S MURDER.

(2) THE FARC MEMBERS BLEW UP A POWER SUBSTATION AND A POWER PYLON IN SAJANDI, LA FONDA CORREGIMIENTO. THE GUERRILLAS ALSO BURNED THREE VEHICLES ON THE PAN-AMERICAN HIGHWAY– A BALBOA HOSPITAL AMBULANCE, A FUEL TRUCK CARRYING 3,000 GALLONS OF GASOLINE, AND A STATION WAGON. A LARGE SECTION OF EL PATIA VALLEY WAS LEFT WITHOUT ELECTRICITY.

(3) THROUGH SEVERAL TELEPHONE CALLS TO THE POPAYAN MEDIA, THE 8TH FARC FRONT DECLARED THE PAN-AMERICAN HIGHWAY A MILITARY TARGET DURING THE NEXT 48 HOURS IN ORDER TO DEMAND THAT THE GOVERNMENT INVESTIGATE AND ARREST THOSE WHO ARE TRULY RESPONSIBLE FOR JARAMILLO'S MURDER.

(4) TRANSPORTATION SERVICES BETWEEN [WORD INDISTINCT] AND PASTO IN SOUTHERN COLOMBIA HAVE BEEN SUSPENDED AS A RESULT OF THE TERRORIST ATTACKS, WHICH BEGAN AT 0300 TODAY.

(5) POLICE AND 3D INFANTRY BRIGADE MEMBERS WERE DEPLOYED TO THE AREAS WHERE THE 8TH FRONT OF THE SELF-STYLED REVOLUTIONARY ARMED FORCES OF COLOMBIA BLOCKED TRAFFIC. THE 8TH FARC FRONT MEMBERS SAID THEIR MILITARY ACTIONS ARE AIMED AT DEMANDING THE CLARIFICATION OF UP PRESIDENTIAL CANDIDATE BERNARDO JARAMILLO'S MURDER.

(6) THE POPAYAN POLICE DEPARTMENT CONFIRMED THAT THE ATTACKS TOOK PLACE ON THE PAN-AMERICAN HIGHWAY AND AT LA FONDA CORREGIMIENTO, PATIA MUNICIPALITY.

(7) GOVERNOR FERNANDO IRACONI SAID THAT THE REGIONAL SECURITY COUNCIL WILL MEET IN POPAYAN IN A FEW MINUTES TO ASSESS THE MOST RECENT TERRORIST ATTACKS AND TO ADOPT THE NECESSARY MEASURES.

Figure 2: A Sample MUC-4 text

murder, so it is important to recognize that his murder is a separate incident. The second paragraph describes the terrorist attacks, which include bombings and burnings in several locations. Whether these bombings and burnings are distinct incidents or a single, collective incident is an interesting question as well. The third and fifth paragraphs again mention both the terrorist attacks and Jaramillo's murder. This text illustrates how even relatively short, cohesive texts can reference multiple events that may be easily confused. Keeping track of the separate incidents requires event tracking to assign each piece of information to the appropriate incident, and consolidation to combine different pieces of information and merge multiple references to the same information.

At first glance, it might appear that discourse analysis is simpler for information extraction than for general story understanding because only certain types of objects and events need to be tracked. But in some ways the problem is more challenging because the discourse analyzer has to do its job without complete knowledge. Most IE systems extract information in a piecemeal fashion so that only relevant text segments are recognized and saved. Consequently, the output representation may not include information that is crucial to distinguish events. For example, consider the first paragraph of the text in Figure 2. An IE system might extract the following pieces of information:

(a) MEMBERS OF THE 8TH FRONT OF THE SELF-STYLED REVOLUTIONARY ARMED FORCES OF COLOMBIA [FARC] HAVE CARRIED OUT TERRORIST ATTACKS

(b) TERRORIST ATTACKS IN SOUTHERN CAUCA DEPARTMENT

(c) PATRIOTIC UNION [UP] PRESIDENTIAL CANDIDATE BERNARDO JARAMILLO OSSA'S MURDER

Given only these sentence fragments, one could easily infer that Jaramillo's murder was one of the terrorist attacks. The key phrase "to protest" would probably not be extracted because protesting is not usually associated with terrorism. It is unlikely that a terrorism dictionary would contain a case frame to recognize this expression. But the phrase "to protest" is essential to understand that the murder happened prior to the attacks reported in the article.

These examples emphasize the fine line between information extraction and general story understanding. It is easy to argue that IE systems will always suffer from a ceiling effect if they operate with tunnel vision and do not process portions of the text that do not appear to be directly relevant to the domain. On the other hand, it would be self-defeating to adopt the attitude that we should not attempt to build IE systems until the natural language understanding problem has been completely solved. There is a wide spectrum of language processing techniques, with shallow text understanding at one end and deep text understanding at the other. Choosing where one wants to sit on this spectrum is a central issue when designing a text analyzer.

Part of the difficulty with discourse processing stems from the fact that it depends heavily on world knowledge. So it is not surprising that one of the emerging trends in information extraction involves developing techniques to automate the acquisition of discourse knowledge. In the next section, we overview the general trend toward automated knowledge acquisition for information extraction systems.

2.3 Automated knowledge acquisition

The first generation of information extraction systems relied on a tremendous amount of hand-coded knowledge. Consider the UMass/MUC-3 system [LCF$^+$91a], which was designed for the MUC-3 terrorism domain. The UMass/MUC-3 system used three primary knowledge bases: a hand-coded lexicon, hand-coded case frames, and hand-coded discourse rules. The lexicon contained over 5000 words which were tagged with parts-of-speech, semantic features, and a few other types of linguistic information. The lexicon was engineered specifically for the terrorism domain, so only tags associated with terrorism were used. For example, the word "party" was tagged as a noun but not as a verb, and its only semantic tag corresponded to the political party word sense and not the celebration sense.

The dictionary of extraction patterns contained 389 case frames designed to extract perpetrators, victims, targets, and weapons. Initially, the case frames represented simple patterns and phrases, but over time some of them because quite complex. For example, the %KIDNAP-CAPTURE% case frame originally represented the expression <**X**> **captured** <**Y**>, where X is extracted as the perpetrator of a kidnapping and Y is extracted as the victim. However, this case frame frequently misfired for events such as police arrests and prisoner escapes. After repeated modifications, the activating conditions for this case frame evolved into a complex function that required X to be a terrorist or an organization, but not a civilian or a law enforcement agent, and required Y not to be a terrorist, a prisoner, or an organization.[2]

Another major component of the UMass/MUC-3 system was a rule base for discourse analysis. The UMass/MUC-3 system was organized as a pipelined system consisting of two main modules: sentence analysis and discourse analysis. The sentence analyzer, CIRCUS [Leh91], performed a shallow syntactic analysis of each sentence and produced instantiated case frames. Each sentence was processed independently of the others. The discourse analyzer was then responsible for putting the pieces back together. Since CIRCUS extracts information in a piecemeal fashion, discourse processing involves not only figuring out how one sentence relates to another, but also how case frames activated in the same sentence relate to one another. The discourse analyzer had many jobs, including coreference resolution (pronouns and noun phrases), case frame merging, event segmentation, and relevance filtering. Domain-specific rules for all of these problems were manually encoded in the system.

The UMass/MUC-3 system was an exercise in manual knowledge engineering. While the system performed well in MUC-3 [LCF$^+$91b], it took a tremendous amount of time, energy, and expertise to make it work as well as it did. And the UMass system was no exception — virtually all of the systems that performed well in MUC-3 required many person-months of manual labor.

So it is not surprising that there is strong interest in developing techniques to acquire the necessary domain-specific knowledge automatically. Several systems have been developed to generate domain-specific extraction patterns automatically or semi-automatically [Huf95, KM93, Ril96a, SFAL95]. There have also been efforts to automate various aspects of discourse processing [AB95, ML95, SL94]. And some researchers have used the information extraction framework to focus on general issues associated with lexical acquisition [Car93, HL94].

In the next section, we present our efforts to automatically generate case frames for information extraction. We describe a system called AutoSlog that was one of the first dictionary construction systems developed for information extraction. AutoSlog generates extraction patterns using a specially annotated training corpus as input. We then explain how AutoSlog evolved into its successor, AutoSlog-TS, which generates extraction patterns without an annotated training corpus. AutoSlog-TS needs only a preclassified text corpus consisting of relevant and irrelevant sample texts. AutoSlog-TS represents a major step toward building conceptual dictionaries from raw text. We have used the case frames generated by AutoSlog-TS for both information extraction and text classification tasks to demonstrate that tools and techniques developed for information extraction can be useful for other natural language processing tasks as well.

3 Automatically Generating Case Frames for Information Extraction

One of the most substantial bottlenecks in building an information extraction system is creating the dictionary of domain-specific extraction patterns. Almost all of the IE systems presented at the message understanding conferences have relied on hand-crafted case frames. Recently, however, there have been several efforts to automate the acquisition of extraction patterns.

Two of the earliest systems to generate extraction patterns automatically were AutoSlog [Ril93] and PALKA [KM93]. More recently, CRYSTAL [SFAL95] and LIEP [Huf96] have been developed. All of these systems use some form of manually tagged training data or user input. For example, AutoSlog requires text with specially tagged noun phrases. CRYSTAL requires text with specially tagged noun phrases as well as a semantic hierarchy and associated lexicon. PALKA requires manually defined frames (including keywords), plus a semantic hierarchy and associated lexicon, and user input is sometimes needed to resolve competing hypotheses. LIEP uses predefined keywords and object recognizers, and depends on user interaction to assign an event type to each relevant sentence.

Defining extraction patterns by hand is time-consuming, tedious, and prone to errors and omissions, so all of these systems represent important contributions to automated dictionary construction. The knowledge-engineering bottleneck has not been eliminated yet, but it has been greatly reduced and simplified. For example, it took approximately 1500 person-hours to build the UMass/MUC-3 dictionary by hand, but it took only 5 person-hours to build a comparable dictionary using AutoSlog (given an appropriate training corpus) [Ril96a]. Furthermore, only minimal expertise is needed to generate a dictionary. Defining case frames by hand requires working knowledge of the domain, natural language processing, and the underlying sentence analyzer, but generating a training corpus requires only knowledge of the domain. In essence, the knowledge-engineering effort has shifted from the hands of NLP specialists to the hands of domain experts, which is more realistic for most applications.

First, we describe the AutoSlog dictionary construction system that creates domain-specific extraction patterns using an annotated training corpus. Next, we describe its successor, AutoSlog-TS, which creates extraction patterns using only raw text. AutoSlog-TS reduces the knowledge-engineering bottleneck even further by eliminating the need for specialized training data. Finally, we describe experimental results with AutoSlog-TS for both information extraction and text classification.

3.1 AutoSlog: generating case frames from annotated text

In retrospect, we realized that the manual dictionary construction effort for the UMass/MUC-3 system was remarkably straightforward. The process generally involved two basic steps:

1. Observe a gap in the dictionary by identifying a noun phrase that should have been extracted but was not.
2. Find a key word in the sentence that identifies the conceptual role of the noun phrase and define a case frame that is activated by that word in the same linguistic context.

For example, consider the following sentence:

THE GOVERNOR WAS KIDNAPPED BY TERRORIST COMMANDOS.

The governor should be extracted as a kidnapping victim, and the terrorist commandos should be extracted as the perpetrators. If either of these noun phrases is not extracted by a case frame, then there must be a gap in the dictionary.

Suppose that the governor is not extracted. Then the key question is what expression should have extracted it? The word “kidnapped” clearly suggests a relevant incident (a kidnapping) so “kidnapped” should trigger a kidnapping case frame. To recognize that the governor played the conceptual role of the victim, the kidnapping case frame must have a victim slot that is filled by the subject of the passive verb “kidnapped.” In essence, the case frame must represent the expression: **<X> was kidnapped**.

Similarly, suppose that the terrorist commandos were not extracted. Again, the word “kidnapped” is the key word that suggests a relevant incident and should trigger a kidnapping case frame. The terrorist commandos should be assigned to the conceptual role of the perpetrator and, since “terrorist commandos” is the object of the preposition “by”, the case frame should represent the expression **was kidnapped by <Y>**.

AutoSlog was designed to mimic this process. As input, AutoSlog needs a set of texts and noun phrases that should be extracted from those texts.[3] The noun phrases must be labeled with their conceptual role and event type. For example, in the previous example “the governor” should be labeled as a *victim* in a *kidnapping* event and the “terrorist commandos” should be labeled as a *perpetrator* in a *kidnapping* event. Figure 3 shows what an annotated text would look like.

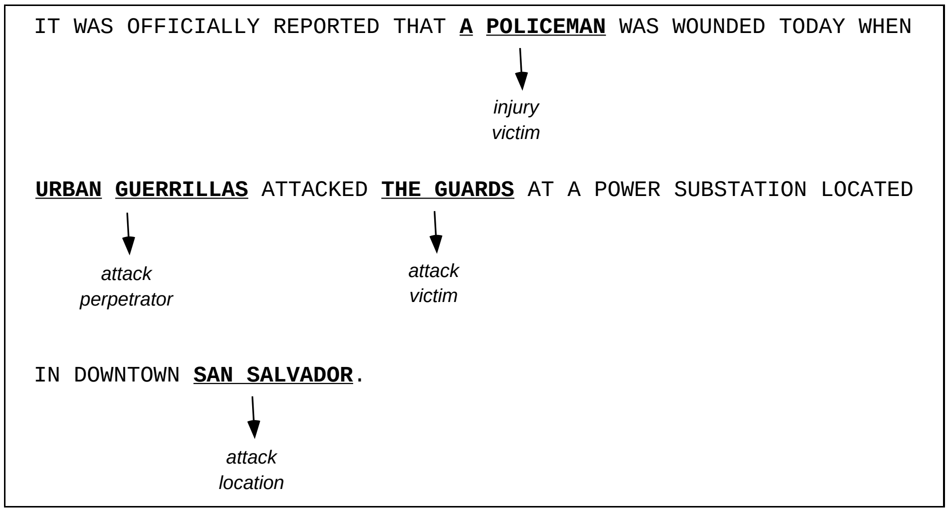
IT WAS OFFICIALLY REPORTED THAT A POLICEMAN WAS WOUNDED TODAY WHEN
injury victim
URBAN GUERRILLAS ATTACKED THE GUARDS AT A POWER SUBSTATION LOCATED
attack perpetrator
attack victim
IN DOWNTOWN SAN SALVADOR.
attack location

Figure 3: An annotated sentence for AutoSlog

Given this training data, AutoSlog generates a case frame to extract each tagged noun phrase. In theory, it would make sense for case frames to extract more than one object, but for simplicity AutoSlog is restricted to single-slot case frames that extract only a single object. AutoSlog uses a small set of heuristic rules to decide what expression should activate the case frame, and from which syntactic constituent the slot should be filled. Figure 4 shows the heuristic rules used by AutoSlog, with examples from the terrorism domain.

Linguistic Pattern	**Example**
<**subject**> **active-verb**	<perpetrator> bombed
<**subject**> **passive-verb**	<victim> was murdered
<**subject**> **verb infinitive**	<perpetrator> attempted to kill
<**subject**> **auxiliary noun**	<victim> was victim
active-verb <**direct-object**>	bombed <target>
infinitive <**direct-object**>	to kill <victim>
verb infinitive <**direct-object**>	threatened to attack <target>
gerund <**direct-object**>	killing <victim>
noun auxiliary <**direct-object**>	fatality was <victim>
noun preposition <**noun-phrase**>	bomb against <target>
active-verb preposition <**noun-phrase**>	killed with <instrument>
passive-verb preposition <**noun-phrase**>	was aimed at <target>

Figure 4: AutoSlog heuristics and examples from the terrorism domain

Given a sentence and a tagged noun phrase, AutoSlog first calls a sentence analyzer, CIRCUS [Leh91], to analyze the sentence syntactically. AutoSlog needs only a flat syntactic analysis that recognizes clause boundaries and identifies the subject, verb, direct object, and prepositional phrases of each clause. So almost any parser could be used. AutoSlog finds the clause that contains the targeted noun phrase and determines whether it was the subject, the direct object, or in a prepositional phrase. The heuristic rules are divided into three sets, depending upon the syntactic type of the targeted noun phrase. The appropriate rules are invoked, and the rule that most closely matches the current context is allowed to fire (i.e., the rule with the longest pattern). For example, consider the following sentence:

The *U.S. embassy* in Bogota was bombed yesterday by *FMLN guerrillas*.

Suppose that *U.S. embassy* was tagged as a bombing target and *FMLN guerrillas* was tagged as a perpetrator. Given *U.S. embassy* as the targeted noun phrase, AutoSlog first calls CIRCUS and determines that *U.S. embassy* is the subject of the sentence. The subject rules are invoked and the pattern <**subject**> **passive-verb** fires. This pattern is matched against the input sentence and a case frame is constructed to represent the expression <**target**> **was bombed**. This case frame is activated by passive forms of the verb "bombed" and extracts the subject as a bombing target. Similarly, given *FMLN guerrillas* as the targeted noun phrase, AutoSlog determines that it was in a prepositional phrase and the prepositional phrase rules are invoked. AutoSlog uses its own simple pp-attachment algorithm to decide where a prepositional phrase should be attached. In this case, *FMLN guerrillas* should attach to the verb "bombed" and AutoSlog generates a case frame to represent the expression **was bombed by** <**perpetrator**>. This case frame is activated by passive forms of the verb "bombed" and extracts the object of the preposition "by" as a bombing perpetrator.

It is important to note that the input consists of a sentence and tagged noun phrases, while the output consists of case frames that represent linguistic expressions. For example, the case frame generated by *U.S. embassy* will be activated by a variety of expressions such as "X was bombed", "X and Y were bombed", "X has been bombed", and "X and Y have been bombed." These words do not have to be adjacent because the natural language processing system uses syntactic information to activate the case frames. For example, the case frame <**target**> **was bombed** will be activated by the sentence below in which the verb ("bombed") and the subject ("Telecom") have many words between them.

TELECOM, COLOMBIA'S SECOND LARGEST TELECOMMUNICATION UTILITY, WAS MERCILESSLY BOMBED YESTERDAY AFTERNOON.

However, many of the case frames generated by AutoSlog will not reliably extract relevant information. AutoSlog can create bizarre or overly general patterns for a variety of reasons, including faulty sentence analysis, incorrect prepositional phrase attachment, or insufficient context. So a person must manually review the case frames and decide which ones are reliable enough for the domain. A simple user interface allows a user to scan each pattern and click an *accept* or *reject* button. The manual review process is very fast; it took a user only five person hours to filter 1237 case frames generated by AutoSlog for the MUC-4 terrorism domain [Ril96a]. The case frames accepted by the user become the final dictionary for the domain.

AutoSlog has been used to generate case frames for three domains: terrorism, joint ventures, and microelectronics [Ril96a]. In the MUC-4 terrorism domain, a dictionary of case frames created by AutoSlog achieved 98% of the performance of the hand-crafted dictionary that achieved good results in the MUC-4 evaluation [Ril93].

AutoSlog was a major contribution toward reducing the knowledge-engineering bottleneck for information extraction systems. Previously, case frames had to be manually defined by people who had experience with natural language processing, the domain, and the sentence analyzer. Using AutoSlog, a dictionary of domain-specific case frames could be constructed automatically, given an annotated training corpus and a few hours of time for manual review.

But there is still a substantial bottleneck lurking underneath AutoSlog: the need for a specially annotated training corpus. Generating the training corpus is both time-consuming and difficult. The annotation process can be deceptively tricky. The user needs to tag relevant noun phrases, but what is a noun phrase? Even simple NPs may include noun modifiers, as in "the heavily armed FMLN guerrillas." Should the user include all of the modifiers, only the relevant modifiers, or just the head noun? Noun phrases can also be quite complex, including conjunctions, appositives, and prepositional phrases. Should the user tag all conjuncts and appositives or just some of them? If not all of them, then which ones? Prepositional phrases can substantially change the meaning of the concept that they modify. For example, "the president of San Salvador" is different from "the president of Colombia." How do you dictate which prepositional phrases are essential and which ones are not? The annotation process can be confusing and arbitrary, resulting in inconsistencies in the tagged training corpus. While the annotation process is less demanding than generating case frames by hand, it is still a major undertaking.

In the next section, we describe the successor to AutoSlog, called AutoSlog-TS, which creates case frames for information extraction without the need for an annotated training corpus. AutoSlog-TS needs only a *preclassified* training corpus consisting of relevant and irrelevant sample texts for the domain.

3.2 AutoSlog-TS: generating case frames from untagged text

The AutoSlog-TS dictionary construction system is designed to create dictionaries of case frames using only a *preclassified* text corpus. As input, AutoSlog-TS needs two piles of texts: one pile of texts that are relevant to the target domain, and one pile of texts that are irrelevant to the domain. Nothing inside the texts needs to be tagged in any way. The motivation behind this approach is that it is relatively easy for a person to generate such a training corpus. The user simply needs to be able to identify relevant and irrelevant sample texts. Anyone familiar with the domain should be able to generate a training corpus with minimal effort. The goal of AutoSlog-TS is to exploit this very coarse level of domain knowledge to generate case frames that represent important domain-specific expressions.

AutoSlog-TS is essentially an exhaustive version of AutoSlog combined with statistical feedback. The dictionary construction process consists of two stages: pattern generation and statistical filtering. Figure 5[4] illustrates this process.

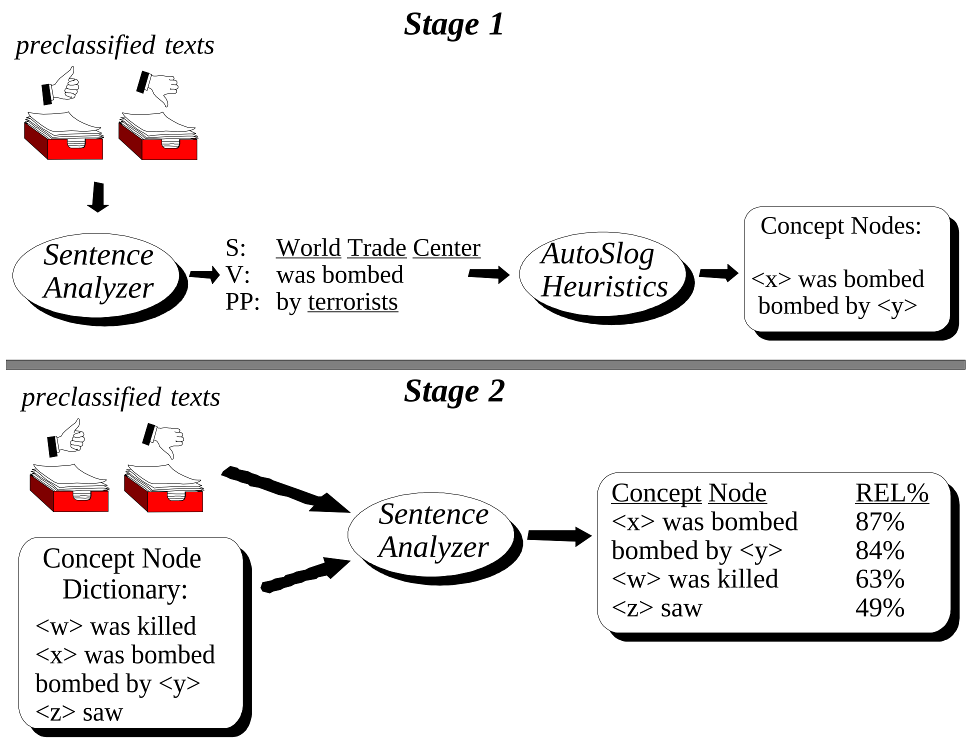

Figure 5: AutoSlog-TS flowchart

In Stage 1, AutoSlog-TS pretends that every noun phrase is a candidate for extraction. A sentence analyzer, CIRCUS, is used to identify all of the simple NPs in a text. For each noun phrase, the heuristic rules shown in Figure 4 are applied to generate extraction patterns. One difference between AutoSlog and AutoSlog-TS is that AutoSlog-TS allows all applicable rules to fire. For example, given the sentence "terrorists attacked the U.S. embassy" and the noun phrase "terrorists", both the <**subject**> **active-verb** and <**subject**> **verb direct-object** rules would apply. Two extraction patterns would be generated: <**perpetrator**> **attacked** and <**perpetrator**> **attacked embassy**.

Allowing multiple rules to fire gives AutoSlog-TS additional flexibility. AutoSlog-TS can create several patterns of varying length to extract the same information. Ultimately, the statistics will reveal whether the shorter, more general pattern has sufficient reliability or whether the longer, more specific pattern is needed. We could have allowed the original AutoSlog system to fire multiple rules as well, but it would have multiplied the number of case frames that had to be manually reviewed. As we will explain shortly, AutoSlog-TS uses statistical information to rank order the case frames so only the most promising case frames need to be reviewed.

The purpose of Stage 1 is to create a giant dictionary of case frames that are literally capable of extracting every noun phrase in the training corpus. In experiments with 1500 texts from the MUC-4 terrorism corpus, AutoSlog-TS created 32,345 unique case frames [Ril96b]. Some of these case frames represent domain-specific patterns, but most of them represent general expressions that are not specific to the domain.

In Stage 2, we use statistical feedback to separate the domain-specific case frames from the general ones. This step takes advantage of the preclassified nature of the training corpus to generate relevance statistics for each case frame. The first step is to load all of the case frames into the sentence analyzer and reprocess the training corpus. Each time a case frame is activated, we record whether it was activated in a relevant text or in an irrelevant text. When the entire training corpus has been reprocessed, we generate final relevance statistics for each case frame. For each case frame, we estimate the conditional probability that a text is relevant given that it activated the case frame. The formula is:

$$\Pr(relevant\ text \mid text\ contains\ case\ frame_i) = \frac{rel-freq_i}{total-freq_i}$$

where $rel - freq_i$ is the number of instances of $case - frame_i$ that were activated in relevant texts, and $total - freq_i$ is the total number of instances of $case - frame_i$ that were activated in the training corpus. For simplicity, we will refer to this probability estimate as a case frame's *relevance rate*.

Note that case frames representing general expressions will be activated in both relevant and irrelevant texts. For example, expressions like "was reported" will appear in a wide variety of texts. If the corpus contains a 50/50 split of relevant and irrelevant texts, then we would expect general case frames to have about a 50% relevance rate. Our approach is based on the intuition that the domain-specific expressions will be much more common in relevant texts than in irrelevant texts so the domain-specific case frames will have higher relevance rates.

After the corpus has been processed a second time, we rank the case frames in order of relevance to the domain. In recent experiments [Ril96b], we used the following ranking function.

$$score_i = relevance\ rate_i * log_2(frequency_i)$$

One exception is that case frames with a relevance rate $\leq$ 0.5 received a score of zero since they were negatively correlated with the domain (the training corpus was 50% relevant). The ranking function gives weight to both a case frame's relevance rate as well as its overall frequency in the training corpus. For example, if a case frame is activated only twice in a large corpus then it is probably not very important. On the other hand, case frames that are activated frequently but have only a moderate relevance rate (say 70%) may in fact be very important to the domain. For example, the expression "X was killed" is a crucial pattern for the terrorism domain because people are often killed in terrorist incidents. But people are also killed in many other types of events so this pattern will appear in irrelevant texts as well. Our ranking function gives preference to case frames with either a high frequency and a moderate relevance rate, or a moderate frequency and a high relevance rate. We do not claim that this is the best possible ranking function, but it seemed to work fairly well in our experiments.

If the ranking function does its job, then the domain-specific case frames should float to the top. The ranking function is crucial because the dictionary of case frames is so large that it would be unreasonable to expect a person to review them all by hand. Once the case frames are ranked, a person can skim the "best" case frames off the top. One of the main advantages of the ranking scheme is that it prioritizes the case frames for manual review so that the strongest case frames are reviewed first, and the user can review those further down on the list as time permits.

We applied AutoSlog-TS to the 1500 MUC-4 development texts [MUC92], of which about 50% were relevant. AutoSlog-TS generated 32,345 unique case frames. Due to memory constraints, we threw away all case frames that were proposed only once since they were not likely to be very important.[5] This left us with 11,225 unique case frames, which we loaded into the system for Stage 2. We then computed the relevance rate for each of these case frames and ranked them using the scoring function just described. The 25 top-ranked patterns (in the list of 11,225 ranked case frames) appear in Figure 6.

1. <subj> exploded	14. <subj> occurred
2. murder of <np>	15. <subj> was located
3. assassination of <np>	16. took_place on <np>
4. <subj> was killed	17. responsibility for <np>
5. <subj> was kidnapped	18. occurred on <np>
6. attack on <np>	19. was wounded in <np>
7. <subj> was injured	20. destroyed <dobj>
8. exploded in <np>	21. <subj> was murdered
9. death of <np>	22. one of <np>
10. <subj> took_place	23. <subj> kidnapped
11. caused <dobj>	24. exploded on <np>
12. claimed <dobj>	25. <subj> died
13. <subj> was wounded	

Figure 6: The Top 25 Extraction Patterns

Most of these expressions are clearly associated with terrorism, so the ranking function appears to be doing a good job of pulling the domain-specific expressions up to the top. The next step is to have a person review the most highly ranked patterns. The manual review process serves two purposes: (1) as confirmation that a pattern is relevant to the domain, and (2) to label the case frame with an event type and a conceptual role for the extracted item. For example, the first case frame in Figure 6, **<subj> exploded**, should be labeled as a bombing case frame that will probably extract an instrument, as in "a car bomb exploded." The second case frame, **murder of <np>**, should be labeled as a murder case frame that will likely extract a victim, as in "the murder of the mayor." The 22nd case frame, **one of <np>**, is an example of a case frame that the user would probably reject as a statistical artifact. Although it may have had a high relevance rate in the training corpus, the expression **one of <X>** is not specific to terrorism and would probably extract too much irrelevant information.

We manually reviewed the top 1970 case frames in the ranked list and retained 210 of them for the final dictionary. The manual review process took approximately 85 minutes. The relatively low number of acceptable case frames is somewhat misleading because the acceptance rate dropped off sharply after the first few hundred. The fast manual review time is also due to the ranking scheme. Most of the good case frames clustered at the top, so after the first few hundred case frames the review process mainly consisted of clicking the reject button to dismiss obviously irrelevant patterns.

3.3 Experiments with Autoslog-TS

AutoSlog-TS was designed to be an extension of AutoSlog that substantially reduces the human effort needed to generate appropriate training texts, while still producing a good dictionary of extraction patterns. Our goal was to show that the dictionaries produced by AutoSlog and AutoSlog-TS achieve comparable performance.

We compared dictionaries created by AutoSlog and AutoSlog-TS for the MUC-4 terrorism domain. Details of this experiment can be found in [Ril96b]. We evaluated both dictionaries by manually reviewing their output on 100 blind texts from the MUC-4 test set. We then computed recall, precision, and F-measure scores for each dictionary. The results appear in Table 1.

	AutoSlog			**AutoSlog-TS**		
Slot	Recall	Prec.	F	Recall	Prec.	F
Perpetrator	.62	.27	.38	.53	.30	.38
Victim	.63	.33	.43	.62	.39	.48
Target	.67	.33	.44	.58	.39	.47
Total	.64	.31	.42	.58	.36	.44

Table 1: Comparative Results

The results show that the AutoSlog dictionary achieved slightly higher recall, and the AutoSlog-TS dictionary achieved slightly higher precision and F-measure scores. Our analysis of the raw data showed that the differences in correct and missing output were not statistically significant even at the $p < 0.20$ significance level, but the difference in spurious output was statistically significant at the $p < 0.05$ significance level. We conclude that there was no significant difference between AutoSlog and AutoSlog-TS in terms of recall, but AutoSlog-TS was significantly more effective at reducing spurious extractions.

While the absolute differences in correct and missing output were not significant, AutoSlog did slightly outperform AutoSlog-TS in recall. One possible explanation is that AutoSlog-TS is at the mercy of the ranking function because a human cannot reasonably be expected to review all of the 11,000+ extraction patterns produced by AutoSlog-TS. The most important domain-specific patterns must be ranked highly in order to be considered for manual review. We believe that there were many good extraction patterns buried deep in the ranked list that did not get reviewed. If so, then AutoSlog-TS is ultimately capable of producing higher recall than AutoSlog.

As further evidence of this, we observed that AutoSlog-TS produced 158 case frames with a relevance rate $\geq$ 90% and frequency $\geq$ 5, but only 45 of them were in the AutoSlog dictionary. Figure 7 shows some of the the case frames that clearly represent patterns associated with terrorism.[6] However, many of these patterns had relatively low frequency counts and were not ranked highly.

We also wanted to demonstrate that AutoSlog-TS could be used for tasks other than information extraction. So we used the case frames generated by AutoSlog-TS for a text classification task. In previous research, we developed a text classification algorithm called the *relevancy signatures algorithm* [RL94] that uses extraction patterns to recognize key phrases for classification. The

was assassinated in X	assassination in X	X ordered assassination
was captured by X	capture of X	X managed to escape
was exploded in X	damage in X	X expressed solidarity
was injured by X	headquarters of X	perpetrated on X
was kidnapped in X	targets of X	hurled at X
was perpetrated on X	went_off on X	carried_out X
was shot in X	X blamed	suspected X
was shot_to_death on X	X defused	to protest X
X was hit	X injured	to arrest X

Figure 7: Patterns found by AutoSlog-TS but not by AutoSlog

task is to automatically classify new texts as relevant or irrelevant to a specific domain. We used the hand-crafted dictionary of extraction patterns in our previous text classification experiments [RL94], but AutoSlog-TS provides a new opportunity to seed the algorithm with a much larger set of potential patterns. The statistics should ultimately decide which ones are most useful for making domain discriminations.

We gave the relevancy signatures algorithm the same set of 11,225 case frames generated by AutoSlog-TS for the terrorism domain, and trained the algorithm using the 1500 development texts from the MUC-4 corpus. We evaluated the algorithm on two blind sets of 100 texts each. On the first test set, the AutoSlog-TS dictionary and the hand-crafted dictionary achieved similar results. On the second test set, however, the AutoSlog-TS dictionary produced several data points with 100% precision while the hand-crafted dictionary did not produce any. Overall, the AutoSlog-TS dictionary performed at least as well as the AutoSlog dictionary. Details of this experiment can be found in [RS95].

In summary, AutoSlog-TS has been used to create a dictionary of case frames for the MUC-4 terrorism domain that performed well on both information extraction and text classification tasks. AutoSlog-TS appears to be at least as effective as AutoSlog at producing useful extraction patterns, although more experiments need to be done with different ranking functions and additional domains. Perhaps most importantly, AutoSlog-TS sharply reduces the amount of manual knowledge engineering required to create a dictionary of domain-specific extraction patterns. AutoSlog-TS is the first system that can generate domain-specific case frames using only raw text as input. So far, these case frames have been applied to information extraction and text classification tasks, but perhaps AutoSlog-TS or similar systems will ultimately be able to produce more complex case frames for other natural language processing tasks as well.

4 Information Extraction and Story Understanding: Bridging the Gap

Research in information extraction has made good progress in recent years, but will it have any impact on computational models of story understanding? Only time will tell, but perhaps some comparisons will shed light on the issue.

For one, both information extraction and story understanding require natural language understanding. This might seem like a trivial statement, but it is important to remember that they are both subject to the same problems of ambiguity, idiosyncracy, and a strong dependence on world knowledge. Research in both fields has tried to minimize these problems by focusing on narrow domains. An information extraction task, by definition, specifies the domain of interest and the types of information which must be extracted. The restricted domain and the focused nature of the task can greatly simplify dictionary construction, ambiguity resolution, and discourse processing. Similarly, story understanding systems usually focus on a single domain or certain aspects of the world in order to minimize the knowledge-engineering effort.

A key difference between information extraction and story understanding is that the latter strives to understand the entire story. These two paradigms represent a tradeoff between depth and scalability. Information extraction systems limit depth by focusing on certain types of information and a specific domain. In exchange for limited depth, IE systems can effectively process a wide variety of texts within the domain. Story understanding systems, on the other hand, aim for deep understanding of whole texts, but usually cannot effectively process arbitrary texts without additional knowledge engineering. One notable attempt to bridge this gap was the FRUMP system [DeJ82], which had its origins in the story understanding community but could be viewed as an early information extraction system that could recognize certain types of events in unrestricted text.

The depth/scalability tradeoff reflects the knowledge engineering bottleneck associated with building natural language processing systems. The growing interest in automated knowledge acquisition should bring the two communities closer together. Information extraction researchers and story understanding researchers could learn a lot from each other. The information extraction community has learned a great deal about robust parsing, case frame generation, and discourse analysis. At least as important is the experience that comes with processing large amounts of real text. Building an IE system is typically an empirical endeavor and IE researchers have come to appreciate that some well-known NLP problems and phenomena do not appear very often in real text, while other problems are much more common than one might expect.

On the other side of the fence, story understanding researchers have a long history of expertise with complex knowledge structures and inference generation. Ultimately, continued progress in information extraction will depend on richer semantics, knowledge structures, and inference mechanisms. Although restricting the domain simplifies many aspects of language processing, it is not a panacea for language understanding. The performance of current IE technology will likely run up against a ceiling effect unless more conceptual knowledge is brought to bear on the problem.

If future IE systems will need to use more conceptual knowledge, where will this knowledge come from? The information extraction paradigm revolves around practical applications and large text collections, so it is imperative that the necessary knowledge can be acquired with little effort. From this vantage point, perhaps the most pragmatic approach is to build gradually upon current IE technology. For example, can extraction patterns be mapped into conceptual primitives? Can combinations of extraction patterns (or primitives) be used to represent more complex knowledge structures? Can automated dictionary construction systems like AutoSlog-TS be used to learn more complex structures? By exploiting the practical nature of IE technology and the insights and theories developed by story understanding researchers, we can entertain the exciting possibility of building increasingly intelligent NLP systems that are practical for large-scale applications. Although the grand challenge of developing a broad-coverage, in-depth natural language understanding system may still be a long way off, an effective synergy between information extraction and story understanding may prove to be a promising starting point for real progress in that direction.

Acknowledgments

Thanks to Claire Cardie for providing helpful comments on an earlier draft of this paper. This work was supported in part by NSF grant MIP-9023174 and NSF grant IRI-9509820.

Notes

[1]An anecdote from an early message understanding conference nicely illustrates this point. Many of the news articles in the corpus mentioned a particular spokesperson, but he was almost always mentioned in texts that were not relevant to the terrorism domain of interest. His name was so common, that one site began using his name as a keyword to identify irrelevant texts. But one day the spokesperson was killed in a terrorist incident, thus becoming a victim himself. From that point on, virtually every text containing his name was relevant.

[2]These activating conditions are domain-specific and would likely be prone to false hits if the case frame was applied to a more general corpus.

[3]In our experiments, AutoSlog used the MUC templates as input, but an annotated training corpus would have been sufficient.

[4]Figure 5 refers to *concept nodes*, which are the case frame structures used by CIRCUS.

[5]AutoSlog often proposes the same case frame multiple times when different noun phrases spawn the same extraction pattern. For example, the case frame representing the expression **murder of <victim>** was proposed in response to many different sentences.

[6]The connected words represent phrases in CIRCUS' lexicon.

References

[AB95] Chinatsu Aone and Scott William Bennett. Applying Machine Learning to Anaphora Resolution. In *Working Notes of the IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*, pages 151–157, 1995.

[Car79] J. G. Carbonell. *Subjective Understanding: Computer Models of Belief Systems*. PhD thesis, Tech. Rept. 150, Department of Computer Science, Yale University, New Haven, CT, 1979.

[Car93] C. Cardie. A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 798–803. AAAI Press/The MIT Press, 1993.

[Cul78] R. E. Cullingford. *Script Application: Computer Understanding of Newspaper Stories*. PhD thesis, Tech. Rept. 116, Department of Computer Science, Yale University, New Haven, CT, 1978.

[DeJ82] Gerald DeJong. An Overview of the FRUMP System. In W. Lehnert and M. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–177. Lawrence Erlbaum Associates, 1982.

[DGCN91] Charles P. Dolan, Seth R. Goldman, Thomas V. Cuda, and Alan M. Nakamura. Hughes Trainable Text Skimmer: Description of the TTS System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 155–162, San Mateo, CA, 1991. Morgan Kaufmann.

[DLW$^+$91] Kathleen Dahlgren, Carol Lord, Hajime Wada, Joyce McDowell, and Edward P. Stabler Jr. ITP: Description of the Interpretext System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 163–170, San Mateo, CA, 1991. Morgan Kaufmann.

[GSM91] Ralph Grishman, John Sterling, and Catherine Macleod. New York University: Description of the Proteus System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 183–190, San Mateo, CA, 1991. Morgan Kaufmann.

[HL94] P. Hastings and S. Lytinen. The Ups and Downs of Lexical Acquisition. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 754–759. AAAI Press/The MIT Press, 1994.

[Huf95] Scott B. Huffman. Learning information extraction patterns from examples. In *Working Notes of the IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*, pages 127–134, 1995.

[Huf96] S. Huffman. Learning information extraction patterns from examples. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 246–260. Springer-Verlag, Berlin, 1996.

[KM93] J. Kim and D. Moldovan. Acquisition of Semantic Patterns for Information Extraction from Corpora. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*, pages 171–176, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[Kol83] J. Kolodner. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7:243–280, 1983.

[LCF$^+$91a] W. Lehnert, C. Cardie, D. Fisher, E. Riloff, and R. Williams. University of Massachusetts: Description of the CIRCUS System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 223–233, San Mateo, CA, 1991. Morgan Kaufmann.

[LCF$^+$91b] W. Lehnert, C. Cardie, D. Fisher, E. Riloff, and R. Williams. University of Massachusetts: MUC-3 Test Results and Analysis. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 116–119, San Mateo, CA, 1991. Morgan Kaufmann.

[Leh81] W. G. Lehnert. Plot Units and Narrative Summarization. *Cognitive Science*, 5(4):293–331, 1981.

[Leh91] W. Lehnert. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In J. Barnden and J. Pollack, editors, *Advances in Connectionist and Neural Computation Theory, Vol. 1*, pages 135–164. Ablex Publishers, Norwood, NJ, 1991.

[LPS86] D. B. Lenat, M. Prakash, and M. Shepherd. CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge-Acquisition Bottlenecks. *AI Magazine*, 6:65–85, 1986.

[ML95] Joseph F. McCarthy and Wendy G. Lehnert. Using Decision Trees for Coreference Resolution. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1050–1055, 1995.

[MSBS91] Christine A. Montgomery, Bonnie Glover Stalls, Robert S. Belvin, and Robert E. Stumberger. Language Systems, Inc.: Description of the DBG System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 171–177, San Mateo, CA, 1991. Morgan Kaufmann.

[MUC91] MUC-3 Proceedings. *Proceedings of the Third Message Understanding Conference (MUC-3).* Morgan Kaufmann, San Mateo, CA, 1991.

[MUC92] MUC-4 Proceedings. *Proceedings of the Fourth Message Understanding Conference (MUC-4).* Morgan Kaufmann, San Mateo, CA, 1992.

[MUC93] MUC-5 Proceedings. *Proceedings of the Fifth Message Understanding Conference (MUC-5).* Morgan Kaufmann, San Francisco, CA, 1993.

[Ril93] E. Riloff. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816. AAAI Press/The MIT Press, 1993.

[Ril96a] E. Riloff. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. *Artificial Intelligence*, 85:101–134, 1996.

[Ril96b] E. Riloff. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049. The AAAI Press/MIT Press, 1996.

[RL94] E. Riloff and W. Lehnert. Information Extraction as a Basis for High-Precision Text Classification. *ACM Transactions on Information Systems*, 12(3):296–333, July 1994.

[RS95] E. Riloff and J. Shoen. Automatically Acquiring Conceptual Patterns Without an Annotated Corpus. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 148–161, 1995.

[SA77] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[SFAL95] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319, 1995.

[SL94] S. Soderland and W. Lehnert. Wrap-Up: A trainable discourse module for information extraction. *Journal of Artificial Intelligence Research (JAIR)*, 2:131–158, 1994.

[Wil78] R. Wilensky. *Understanding goal-based stories*. PhD thesis, Tech. Rept. 140, Department of Computer Science, Yale University, New Haven, CT, 1978.